RemindMe: Plugging a Reminder Manager into Email for Enhancing Workplace Responsiveness

Casey Dugan, Aabhas Sharma, Michael Muller, Di Lu, Michael Brenndoerfer, Werner Geyer

RemindMe: Plugging a Reminder Manager into Email for Enhancing Workplace Responsiveness

Casey Dugan*, Aabhas Sharma*, Michael Muller*, Di Lu**, Michael Brenndoerfer***, Werner Geyer*

*IBM Research, Cambridge MA USA; **University of Pittsburgh PA USA; ***IBM, Rueschlikon, Switzerland

{cadugan, michael_muller, Werner.Geyer}@us.ibm.com, Aabhas.Sharma@ibm.com, dil16@pitt.edu, BRE@zurich.ibm.com

Abstract. Reminding others to do something or bringing something to someone's attention by sending reminders is common in the workplace. Our goal was to create a system to reduce the cognitive overhead for employees to manage their email, specifically the incoming and outgoing requests with their colleagues and others. We build on prior research on social request management, interruptions, and cognitive psychology in the design of such a system that includes an email reminder creation algorithm, with a built-in learning mechanism for improving such reminders over time, and a reminder delivery user interface. The system is delivered to users through a browser plugin, allowing it to be built on top of an existing web-based email system within an enterprise.

Keywords. reminders • email • request management • browser plugin

1 Introduction

Email is used for many purposes [13, 30] and the burden of managing it is a core part of the work environment [8, 13, 20, 44]. This is particularly true as many use email for task management [4, 5, 14, 20, 34, 40]. In this paper, we consider tasks represented in incoming and outgoing requests in email in the work place. While such requests can occur in many kinds of channels (real-time messaging, social collaboration platforms etc.), recent research has shown that email is still the predominant channel for requests in the workplace [34]. We have observed that a common strategy in managing requests in the workplace involves reminding others to do something or bringing something to someone's attention by sending reminders. However, this involves significant cognitive overhead for employees: *I asked Bob for that file a couple of days ago, is it too soon to remind him to send it?* As such, we sought to create a system that proactively reminds users of such request-based emails that require their attention – before their colleagues need to.

A significant challenge in attempting to build such a system is that the very act of reminding users could distract them from their tasks at hand. Most studies of reminding have involved the unsolicited presentation of a message -- i.e., an "interrup-

tive notification" [36]. In workplaces, notifications interrupt on-going work, and are therefore stressful [33] and can result in negative emotional experiences of work [22, 36]. While researchers have studied the impact of interruptions [7] and recovery from interruptions [37] as a matter of cognitive load and task boundaries [22], there is also evidence that people prioritize interruptions in terms of content and social relationships [16]. Despite the emotional and cognitive costs, users in organizations continue to prefer to receive interruptions, because those notifications provide awareness of others' work and needs [23].

In our work, we adopt an unobtrusive way of notifying users. Instead of interrupting the user's on-going work, we provide a slowly-changing, ambient series of non-interruptive notifications. These are presented in a dedicated region, directly within the context of a user's mail client. This allows the user to be aware of such pending tasks, while being in control of the decision of whether the interruption is important enough to act upon right now.

Another significant challenge in creating such a system is determining which emails a user should be reminded about. While it is relatively easy to classify emails into binary categories such as spam-vs.-ham [15], or single dimensions such as sentiment [6], the automated analysis of emails to determine which contain requests, or actions a receiver must take, has proven more challenging. Some researchers have explored classifying emails and email-like messages into categories [17]. Early work analyzed rule-based classification of messages similar to emails [31]. Increasingly, researchers are using machine-learning methods to classify emails in general (e.g., [1, 15], for activities [12], for meeting requests [26], and for specific purposes such as prioritization [45]. The most relevant work in this area is predicting the likelihood of a user responding to a given email [11], or the end of a conversation [29]. While such work discusses how a reply prediction algorithm could be used to make users aware of emails they need to reply to (such as through annotating received emails), they stop short of proposing a reminder system. We believe this comes back to balancing the required reply behavior (as opposed to expected) against the cost of the interruption. For example, just because I typically reply with "Thanks" when a colleague fulfills my requests, the system likely shouldn't remind me to do so a week later if I forgot.

To address these challenges, we report on a system architecture for RemindMe, which supports identifying, presenting and acting on reminders from within complex email interfaces, as part of a longer-term goal of providing automated support to help users manage their requests and reminders. Our envisioned systems would be similar in spirit to the opportunistic agents of Dey and Abowd [10] and Myers and Yorke-Smith [35], but would focus on *social reminders* among collaborating colleagues. This paper constitutes a first step, which solves the problem of managing social reminders. It takes a novel approach in delivering such reminders to users, balancing interruption and awareness. It builds on prior machine learning work in determining which emails likely require responses. And it incorporates mechanisms to gather user feedback, necessary for the algorithm to be able to learn the user's preferences towards the nature of requests that they would like to be reminded about and better identify when to remind users about such requests.

2 Context, Challenges, and Technical Approach

Enterprise users face challenges with email overload [29, 44] and social request management [34], and are most likely to need proactive reminding capabilities. To help them, we designed and built RemindMe within the context of IBM.

In evaluating how to build the reminder system, context, pluggability and generalizability were taken into account. Reminding users within the context of a mail application was important to ensure they were currently focused on their regular email activity and minimize the interruptions of reminders [26]. Pluggability focused on the ability to extend an existing IBM email tool, rather than building a new one. Generalizability related to building the tool in a way to allow for testing with users from other organizations with minimal development costs.

To support reminding within an existing mail application context, we designed our solution to extend IBM's popular web-based email client, through the use of a browser plugin. As Firefox was the most popular browser used to access the mail platform within IBM, we first focused on a Firefox plugin.

The plugin format also helped with the generalizability of the solution. We could give selected users the plugin to install for testing purposes, who could uninstall it if they chose to. We also gained significant control over modifying the existing mail interface, and the ability to modify the plugin to support different web-based email systems supporting generalizability. The downsides of this approach are that our system couldn't automatically be rolled out to all IBM users simultaneously (since it requires installation), we support only Firefox browser users, and we are only able to collect data and remind users while they use the mail site within their browser.

While the Browser Plugin focuses on the interface and limited, light-weight analysis, a server component is required to conduct more computationally intense analysis including storage and processing of a user's email, request identification, and reminder creation, storage, and feedback.

As shown in Figure 1, the ReminderMe system consists of three overall modules. The first module is responsible for generation of Reminders. The email synchronization is responsible for collecting the user's emails and is the only part of

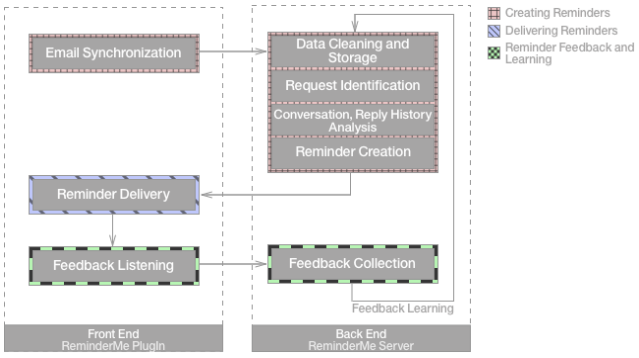

Figure 1. Flowchart depicting interactions of the different system components

this module that operates within the plugin. The mail data collected is then analyzed on the RemindMe server, which also generates the reminders.

The second module is responsible for delivering the system-generated reminders to the user; it operates entirely within the plugin. Finally, the third module is responsible for obtaining feedback from the user, analyzing the information and providing this to the Reminder creation module in the form of feedback learning. This module is divided across both the RemindMe plugin and back-end server.

3 Creating Reminders

3.1 Email Synchronization

As part of the reminder creation process, email data must first be collected from the underlying enterprise mail system and sent to the RemindMe server for storage and processing (Figure 1). When the user authenticates via the email service's web interface, the browser activates the RemindMe plugin. The plugin then continuously queries the same mail retrieval REST APIs used to populate the mail interface and sends the results to the RemindMe server for synchronization. Polled emails include both incoming and sent emails. We periodically re-poll to check for changes such as deletes and changes to state (such as unread/read).

3.2 Data Cleaning and Storage

Emails arriving at the server are stripped of HTML and rich text, reply histories (in the case of replies to threads), and email signatures. New line whitespace is preserved, resulting in an optimized line-based representation of the textual content (both subject and body) of the emails. The procedure is similar to [9].

3.3 Request Identification

Within the RemindMe system, we chose to focus on reminding users about emails containing requests, rather than focusing on other features related to reply prediction [11]. To identify such requests within email messages, we use a third-party "action identification" API from Watson Workservices, found at https://api.watsonwork.ibm .com. This is a machine-learning classifier that analyzes provided text and returns a set of identified requests, questions etc. (such as "please send the file"), the position in the text where these were found, confidence scores etc. We run this identification on each textual content line from the body of an email, as well as the subject.

3.4 Conversation Analytics, User-Reply-History Analysis

In addition to the storage of individual emails, we also store a representation of overall conversation threads, which tracks which emails are replies to other emails (e.g.,

[24, 41]). For each RemindMe user we also calculate statistics on their reply patterns with other users, similar to [29]. These statistics include how often they receive emails from another user and reply to those emails, as well as how long it takes them to reply to those emails on average.

3.5 Reminder Creation

Users are frequently overwhelmed with the volume of work items to manage [3, 43], email in general [29, 44], and email-requests in particular [28, 34]. However, users are reluctant to do extra work to mark emails as needing reminders [4]. Therefore, in RemindMe, we try to automatically determine whether a user is likely to need a reminder about a given email in the future. To do so, the results of the Request Identification analysis described above are used to determine the likelihood that a given message contains a request (based on the subject or body of the email including questions, requests, etc., with a probability score above a certain threshold, see [11] for a related approach). However, in practice, request identification alone is not sufficient, because certain emails users are unlikely to reply to were found to have correctly identified requests, such as spam emails asking "Have you checked your credit score?" Therefore, we also analyze the reply-statistics for a given user to determine how likely they are to reply to this particular user. If they have received many emails, but replied to none, a reminder is not created. In addition to automatically generating reminders, reminders are also created for emails a user has manually flagged as requiring follow-up. However, as only 5% of IBM users actively use this feature of the underlying mail system (see also [4]), most reminders are automatically determined.

For each reminder created, we need to determine when it should be shown. As an initial approximation of this, we again make use of the reply history between user *dyads* – i.e., a sending and receiving user (see also [29]), to determine the average reply time for the message receiver. If a reply to this message is not processed by the system by the time this period has elapsed, a reminder about the message is activated for the user (Figure 1). This works in both directions, for requests a user has sent to others where replies have not been received as well as requests sent to the user.

Finally, for each reminder, we need to determine what information from the original email should be included as part of the reminder message. To do so, we draw upon the significant body of research in the field of cognitive psychology that studied the effect of providing reminders on the performance of prospective memory (or a person's capability to remember to do an activity in the future) since the 1990s. Guynn et al. [18] found that the most effective reminders referred both to the information about the trigger event of the task and to the intended activity. More recently, Baldwin [2] showed that text-only reminders had a limited period of impact, which could be extended by including a picture implying the task. As such, for each reminder, in addition to storing a reference to the original request email and when the reminder should be activated, we store a "trigger event" description (i.e. "Bob made a request to you in an email with subject 'Customer Briefing' on May 1st"), "intended activity" text determined from an identified request within the email with the highest

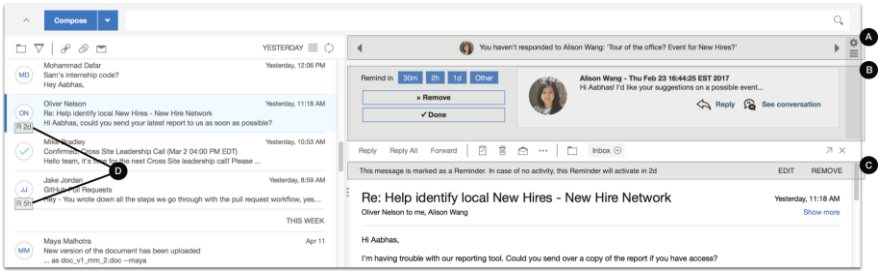

Figure 2. RemindMe browser plugin injects interface components into existing mail application, including: A. Minimal active reminder view. B. Maximized view. C. Contextual reminder information for a selected email. D. Reminder annotations on list views

confidence score (i.e. "Send the file"), as well as a photo of the related person (i.e. the sender if the user is the recipient of the request).

4 Delivering Reminders

While most prior reminder systems have operated through intrusive alerts [26], a key design consideration for RemindMe was to deliver reminders in a fashion that grabs the user's attention, but not enough to distract them from their other email activity, allowing users to make their own decisions about potential interruptions. The interface attempts to strike this balance by creating a dedicated visual 'reminder area,' which is injected by the browser plugin directly above the email viewing space, as shown in Figure 2. This area presents the user with exactly one reminder at a time, to prevent reminder overload. Reminders determined by the server to be currently active then appear in this area and the plugin continuously polls the server for newly activated reminders.

The interface automatically transitions from the current reminder to the next active reminder after a set time, to provide ambient awareness, avoiding additional notification overload. However, this automated transition between active reminders can be personalized through user settings (a popup accessed through the Settings icon shown on the right of Figure 2A). The user can toggle between automatic scrolling vs. manual scrolling. Additionally, the user can manually transition from the currently shown reminder to other active reminders by using two on-screen arrow keys.

The dedicated visual reminder area begins in a "minimized" state, or the area shown in Figure 2A, which includes only the "trigger event" description and photo of the user associated with the reminder, as described in Section 3.5. This is done to minimize the area used by the reminder, which takes away space devoted to reading messages within the mail interface. However, if the user chooses to see more information about the current reminder or interact with it in some way, s/he can click on it to expand the reminder space, to include the area shown in Figure 2B. The maximized view of the reminder display region includes the "intended activity" text described above as well as options for facilitating replies and providing other feedback on the

reminder. The user has the option of viewing the entire conversation thread related through a "See Conversation" button. A "Reply" button also enables the user to write their response to original reminder-triggering email directly from the RemindMe plugin interface. Choosing either of these options causes the relevant content to open up in a new tab rather than disturbing the current state of the email interface.

5 Reminder Feedback & Learning Mechanisms

As reminder creation in RemindMe involves various aspects of machine learning (learning user reply behaviors, request identification etc.), we have designed the system to collect, store, and learn from both implicit and explicit user feedback on these reminders (Figure 1). Three of the four user interface regions highlighted in Figure 2 include feedback mechanisms for the identification of emails requiring reminders as well as the selection of the appropriate time to present a reminder.

The maximized area shown in Figure 2B addresses currently active reminders. Here, the user can give feedback on both the validity as well as the timing of the reminder. For example, clicking the "Reply" button in this area informs the RemindMe server a request was completed, which deactivates the reminder and removes it from this area. While both the "Done" and "Remove" buttons trigger the removal of the current reminder, they provide different feedback. With the "Done" option, the user provides feedback that the identified request was valid, they have simply already completed it. This feedback tells the system that more work may be necessary to automatically identify completion of requests like these in the future. The "Remove" button, in contrast, provides feedback to the RemindMe server that the current reminder was misidentified. This feedback can be used to tune a personalized machine learning classifier of messages that should be treated as reminders for this user.

The final functionality of Figure 2B is a set of "snooze" options that allow the user to simultaneously give feedback to the system that this was a valid reminder given at the wrong time, while also enabling the user to reschedule it for a better time. The interface provides four options to snooze a currently active reminder: fixed durations of thirty minutes, two hours and one day, and a user-adjustable interval.

The areas shown in Figure 2D and 2C present alternate methods of making users aware of emails which are likely to trigger reminders in the future. The area in Figure 2D presents annotations in the inbox while the area in Figure 2C presents additional reminder details about the currently viewed email. As shown in Figure 2D, in the email list, the plugin adds a small annotation next to messages that are classified as future reminders. The annotation contains the estimated time within which the reminder will activate. This indication allows the user to identify and view an email of interest in a more direct fashion as well as gain early awareness of future reminding activity. As shown in Figure 2C, when a given email is displayed, an added reminder bar affords the user the ability to view, modify and give feedback on whether the respective reminder is active, scheduled for the future, or does not exist. If the current email is not marked as a reminder, the user can manually schedule a future reminder for it, which acts as feedback to the system of a potentially missed request. Should the

current email be marked as an upcoming reminder, the user can view the estimated time within which this email would enter the user interface as an active reminder and manually modify that activation time if necessary. Again, by doing so, the user informs the RemindMe server of a more suitable time to present the relevant reminder. This feedback can serve as ground truth to tune a different machine-learning algorithm that is focused on personalized time intervals.

6 Conclusion

We designed and built a novel proactive reminder system that helps enterprise users cope with work requests buried in email. Our work is informed through previous research [3, 4, 5, 11, 14, 19, 20, 26, 28, 30, 31, 42, 43, 44] and, to the best of our knowledge, describes the first system design that combines various components and algorithms together into an integrated system that tackles "email overload" [44] from a reminder perspective, while balancing between intrusive notifications [26] and ambient information. We presented an architecture that allowed the system to be integrated into an existing web-based mail application through the use of a browser plugin. The interface design was informed by prior research on social request management and interruptions, as well as work in the area of cognitive psychology on the effect of types of reminder information on prospective memory [18, 21].

While our system incorporates prior research and functions as an end-to-end request management tool, there is room for improvement in future work. The system currently uses a simplistic model of request "fulfillment" - it assumes a reply to a given email negates the need for a reminder. However, we understand that in practice the process may be much more complex (i.e. a user replies that they will send a file at a later date, rather than sending the file). As such, we plan to incorporate a more detailed method of determining if a given reply satisfies the request from a previous email, using a method as described in [25]. Another possible improvement to the system could come in the form of sensing the user's current task or task boundaries, such as in [22,23], in choosing when to remind them about certain emails.

Early feedback collected on the system has already established a need to expand on how users can respond to email requests through the reminder interface. For example, a feature can be added through which the system helps the user compose a reminder message. A user also asked for a feature to delay request response delivery until the user's average reply time to the request sender has been reached. A detailed user study is planned to test the personalized learning mechanisms related to reminder identification and timing described in the Reminder Feedback section. The user study would also uncover additional user needs and preferences in using such a system, such as any burden of giving manual feedback on accuracy of timing and content of reminders, and users' privacy requirements in the automated analysis of their email. Finally, through the current system design and through above mentioned improvements, we believe such a reminder system can change how we interact with email in the future.

7 References

1. Alberts, I. and Andre Vellino, A. The importance of context in the automatic classification of email as records of business value: A pilot study. *Proc. AIST*, art. 113. (2013)
2. Baldwin, M. *The effects of reminder distinctiveness and anticipatory interval on prospective memory*. MS Thesis, Clemson University. (2014)
3. Barreau, D., and Nardi, B.A. Finding and reminding: File organization on the desktop. *SIGCHI Bul. 27*(3), 39-43. (1995)
4. Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D.G., and Ducheneaut, N. What a to-do: Studies of task management towards the design of a personal task list manager. *Proc. CHI 2004*, 735-742. (2004)
5. Bellotti, V., Ducheneaut, N., Howard, M., and Smith, I. Taking email to task: The design and evaluation of a task management centered email tool. *Proc.CHI 2003*, 345-352. (2003)
6. Bogawar, P.S., and Bhoyar, K.K. Soft computing approaches to classification of emails for sentiment analysis. Proc. ICIA 2016, art. 22. (2016)
7. Bogunovich, P. and Salvucci, D. The effects of time constraints on user behavior for deferrable interruptions. *Proc. CHI 2011*, 3123-3126. (2011)
8. Bota, H., Bennett, P.N., Awadallah, A.H., and Dumais, S.T. Self-Es: The role of emails-to-self in personal information management. *Proc. CHIIR 2017*, 205-214. (2017)
9. Carvalho, V.R., and Cohen, W.W. Learning to Extract Signature and Reply Lines from Email. *Proc. Conf. Email Anti-Spam*. (2004)
10. Dey, A.K., and Abowd, G.D. Cybreminder: A context-aware system for supporting reminders.*Proc. HUC 2000*, 172-186.
11. Drezde, M., Brooks, T., Carroll, J., Magarick, J., Blitzer, J., and Pereira, F. Intelligent email: Reply and attachment prediction. *Proc. IUI 2008*, 321-324. (2008)
12. Drezde, M., Lau, T., and Kushmerick, N.. 2006. Automatically classifying emails into activities. *Proc. IUI 2006*, 70-77. (2006)
13. Ducheneaut, N. and Bellotti, V.. 2001. Email as habitat. *Interactions 8*(5), 30-38. (2001)
14. Flores, F., Graves, M., Hartfield, B., and Winograd, T. 1988. Computer systems and the design of organizational interaction. *ACM TOIS 6*(2), 153-172. (1988)
15. George, P., and Vinod, P. Machine learning approaches for filtering spam emails. *Proc. SIN 2015*, 271-274. (2015)
16. Grandhi, S. Human interruptability: A relational perspective. *Proc. GROUP 2007*, art. 2. (2007)
17. Grbovic, M., Halawi, G., Karnin, Z., and Maarek, Y. How many folders do you really need? Classifying email into a handful of categories. *Proc. CIKM 2014*, 869-878. (2014)
18. Guynn, M.J., McDonald, M.A., and Einstein, G.O. Prospective memory: When reminders fail. *Mem. & Cog. 26*(2), 287-298.
19. Gwizdka, J. Email task management styles: The cleaners and the keepers. *Proc. CHI 2004*, 1235-1238. (2004)
20. Gwizdka, J. Reinventing the inbox –Supporting the management of pending tasks in email. *Proc. CHI 2002*, 550-551. (2002)
21. Gwizdka, J. Supporting prospective information in email. CHI EA 2001, 135-136. (2001)
22. Iqbal, S.T., and Bailey, B.P. Effects of intelligent notification management on users and their tasks. *Proc. CHI 2008*, 93-102. (2008)
23. Iqbal, S.T., and Horvitz, E. Notifications and awareness: A field study of alert usage and preferences. *Proc. CSCW 2010*, 27-30. (2010)
24. Joty, S., Carenini, G., Murray, G., and Ng, R.T. Exploiting conversation structure in unsupervised topic segmentation for emails. *Proc. EMNLP 2010*, 388-398. (2010)

25. Kalia, A., Nezhad, H.R.M., Bartolini, C., and Singh, M. Identifying business tasks and commitments from email and chat Conversations. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.643.5660&rep=rep1&type=pdf (2013)
26. Kamar, E., and Horvitz, E. Jogger: Models for context-sensitive reminding. *Proc. AAMAS 2011*, 1089-1090. (2011)
27. Kaptein, M. and Halteren, A.. Adaptive persuasive messaging to increase service retention: Using persuasion profiles to increase the effectiveness of email reminders. *Pers. Ubiq. Comp. 17*(6), 1173-1185. (2013)
28. Karger, D.. Creating user interfaces that entice people to manage better information. *Proc CIKM 2011*, 1. (2011)
29. Kooti, F., Aiello, L.M., Grbovic, M., Lerman, K., and Mantrach, A. Evolution of conversations in the age of email overload. *Proc. WWW 2015*, 603-613. (2015)
30. Mackay, W.E. More than just a communication system: Diversity in the use of electronic mail. *Proc. CSCW 1998*, 344-353. (1998)
31. Mackay, W.E., Malone, T.W., Crowston, K., Rao, R., Rosenblitt, D., and Card, S.K. How do experienced information lens users use roles? *Proc. CHI 1989*, 211-216. (1989)
32. Mandic, M., and Kerne, A. Using intimacy, chronology, and zooming to visualize rhythms in email experience. CHI EA 2005, 1617-1620. (2005)
33. Mark, G., Gudith, D., and Klocke, U. The cost of interrupted work: More speed and stress. *Proc. CHI 2008*, 107-110. (2008)
34. Muller, M., Dugan, C., Brenndoerfer, M., Monroe, M., and Geyer, W. What did I ask you to do, by when, and for whom? Passion and compassion in request management. *Proc. CSCW 2017*, 1009-1023. (2017)
35. Myers, K. and Yorke-Smith, N. Proactive behavior of a personal assistive agent. *Proc AAMAS 2008*. (2008)
36. Paul, C., and Komlodi, A. Emotion as an indicator for future interruptive notification experiences. *CHI EA 2012*, 2003-2008. (2012)
37. Salvucci, D.D. On reconstruction of task context after interruption. *Proc. CHI 2010*, 89-92. (2010)
38. Scupelli, P., Kiesler, S., Fussell, S.R., and Chen, C. 2005. Project view IM: A tool for juggling mutiple projects and teams. *Proc. CHI 2005*, 1773-1776. (2005)
39. Singh, N., Tomitsch, M., and Maher, M.L. A time and place for preparatory methods in email. *Proc. CHINZ 2013*, Art. 4. (2013)
40. Siu, N., Iverson, L., and Tang, A. Going with the flow: Email awareness and task management. *Proc. CSCW 2006*, 441-450. (2006)
41. Thomas, G., Zahm, M., and Furcy, D. Using a sentence compression pipeline for the summarization of email threads in an archive. *J. Comp. Sci. Coll.31*(2), 72-78. (2015)
42. Venoglia, G., Dabbish, L., Cadiz, J.J., and Gupta, A. Supporting email workflow. http://research.microsoft.com/en-us/groups/coet/01-88.pdf. (2001)
43. Whittaker, S., Jones, Q., Nardi, B., Creech, M., Terveen, L., Isaacs, E., and Hainsworth, J. ContactMap: Organizing communication in a social desktop. *ACM TOCHI 11*(4), 445-471. (2004)
44. Whitaker, S., and Sidner, C. Email overload: Exploring personal information management of email. *Proc. CHI 1996*, 276-283. (1996)
45. Yoo, S., Yang, Y., and Carbonell, J. 2011. Modeling personalized email prioritization: Classification-based and regression-based approaches. *Proc. CIKM 2011*, 729-738. (2011)